;login:

The USENIX Association Newsletter

Volume 12, Number 3 **May/June 1987**

CONTENTS

The closing date for submissions for the next issue of *;login:* is June 26, 1987

USENIX THE PROFESSIONAL AND TECHNICAL
UNIX® ASSOCIATION

NOTICE

;login: is the official newsletter of the USENIX Association, and is sent free of charge to all members of the Association.

The USENIX Association is an organization of AT&T licensees, sub-licensees, and other persons formed for the purpose of exchanging information and ideas about UNIX[†] and similar operating systems and the C programming language. It is a non-profit corporation incorporated under the laws of the State of Delaware. The officers of the Association are:

President	Alan G. Nemeth
Vice-President	Deborah K. Scherrer
Secretary	Waldo M. Wedel
Treasurer	Stephen C. Johnson
Directors	Rob Kolstad Marshall Kirk McKusick John S. Quarterman David A. Yost
Executive Director	Peter H. Salus

The editorial staff of *;login:* is:

Editor and Publisher	Peter H. Salus usenix!peter
Technical Editor	Kevin Baranski-Walker usenix!kevin
Copy Editor	Michelle Dominijanni {masscomp,usenix}!mmp
Production Editor	Tom Strong usenix!strong

Other staff members are:

Betty J. Madden	Office Manager
Emma Reed	Membership Secretary
Jordan Hayes	Technical Consultant

USENIX Association Office
P.O. Box 2299
Berkeley, CA 94710
(415) 528-8649
{ucbvax,decvax}!usenix!office

Judith F. DesHarnais Conference Coordinator

USENIX Conference Office
P.O. Box 385
Sunset Beach, CA 90742
(213) 592-3243 or 592-1381
{ucbvax,decvax}!usenix!judy

John L. Donnelly Tutorial & Exhibit Manager

USENIX Exhibit Office
Oak Bay Building
4750 Table Mesa Drive
Boulder, CO 80303
(303) 499-2600
{ucbvax,decvax}!usenix!johnd

Contributions Solicited

Members of the UNIX community are encouraged to contribute articles to *;login:*. Contributions may be sent to the editors electronically at the addresses above or through the U.S. mail to the Association office. The USENIX Association reserves the right to edit submitted material.

;login: is produced on UNIX systems using *troff* and a variation of the –me macros. We appreciate receiving your contributions in *n/troff* input format, using any macro package. If you contribute hardcopy articles please send **originals** and leave left and right margins of 1" and a top margin of 1½" and a bottom margin of 1¼".

Acknowledgments

The Association uses a SUN[‡] 3/180S running SUN OS for support of office and membership functions, preparation of *;login:*, and other Association activities.

Connected to the SUN is a QMS Lasergrafix[*] 800 Printer System donated by Quality Micro Systems of Mobile, Alabama. It is used for general printing and draft production of *;login:*.

This newsletter is for the use of the membership of the USENIX Association. Any reproduction of this newsletter in its entirety or in part requires written permission of the Association and the author(s).

[†]UNIX is a registered trademark of AT&T.

[‡]SUN is a trademark of Sun Microsystems, Inc.
[*]Lasergrafix is a trademark of Quality Micro Systems.

 Copyright © 1987 USENIX Association–All Rights Reserved

The RIACS Mail System

Matt Bishop
Research Institute for Advanced Computer Science
NASA Ames Research Center
Moffett Field, CA 94035

ABSTRACT

This document describes the configuration of the RIACS mail system, and how the mail configuration files implement this design. The rationale behind the design, as well as the configuration files, is discussed. The paper concludes with a description of features that may be used to debug the configuration files, and some basic guidelines on how to trace problems with configuration files that have been installed.

1. Introduction

Users who send or receive letters over an electronic network deal with only a small part of the mail system. Typically, sending mail involves several levels of services and protocols, summarized in Figure 1. (This model is called *the MHS model* and is from ISO X400.)

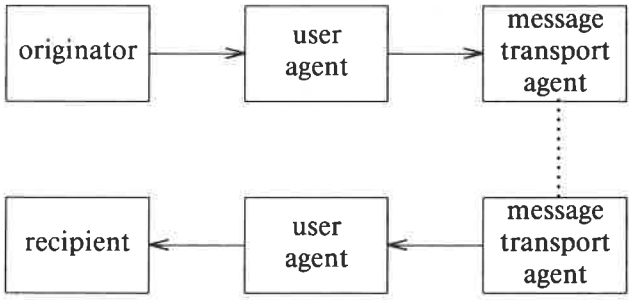


Figure 1. The MHS model.

In this figure, the person who wants to send the letter is called the *originator*. He uses a *user agent*, which is a program that enables him to type and format the letter. The user agent then hands the letter to a *message transport system*. This system is composed of one or more *message transport agents*; each agent accepts the message, determines if the recipient is on its machine, and if so passes the message to the appropriate user agent. If not, it determines which machine it should deliver the message to, and passes the letter on to the message transport agent on that machine. Finally, the letter reaches a user agent on the recipient's machine, and the user agent enables the recipient to read and possibly reply to the letter.

RIACS has many user agents available, ranging from the simple *binmail*(1) [UPM84] to the highly sophisticated *MH Mail Handling System* [ROSE86]. This manual leaves the description of the user agents to other documents; instead it concentrates on the message transport agents.

The next section describes the configuration of the machines at RIACS; the sections after it proceed line-by-line through the gateway and non-gateway *sendmail* configuration files. The final sections discuss how to use *sendmail* [ALLM84a, ALLM84b] to debug a configuration file, and how to install new configuration files. These last sections assume the reader is familiar enough with configuration files that the notation and general layout need not be reviewed; if not familiar with these, the reader should have a copy of [ALLM84b] readily available when reading these sections.

2. RIACS Mail Configuration

The RIACS environment is quite heterogeneous. It consists of a VAX-11/730† which serves as a gateway, a Sequent Balance 21000, an Intel Hypercube, a Ridge 32/V, a Silicon Graphics IRIS 3500, and several Sun 3s, all running some version of UNIX, some sharing file systems using NFS and others not. The following table shows the names of these machines:

Table 1. RIACS Computers and their Names

host name	*computer*
icarus	VAX-11/730
hydra	Sequent Balance 21000
cube	Intel Hypercube
daedalus	Ridge 32/V
pegasus	Silicon Graphics IRIS 3500
lavalite	Sun 3 (has file system)
miranda	Sun 3 (has file system; server)
phun	Sun 3 (client sharing *miranda*'s file system)
zeus	Sun 3 (client sharing *miranda*'s file system)
clavier	Sun 3 (client sharing *miranda*'s file system)
dora	Sun 3 (client sharing *miranda*'s file system)

There are also an Evans and Sutherland P300 graphics system, two Apple Macintoshes, and an IBM PC used for financial matters.

Of these machines, *icarus*, *hydra*, *daedalus*, and the Suns are the only ones which allow the full range of mailing facilities. *Cube* allows local mail to be sent, but does not have any facility for sending mail to another machine except via *uucp*. *Pegasus* does not allow any nonlocal mail to be sent; it, the Evans and Sutherland, the Macintoshes, and the IBM PC do not interact with the mail system.

There are a number of configurations that allow mail to be sent to nonlocal or remote machines. One is to require each host to have all the information needed, so when a letter is sent to *csl.sri.com* from *hydra*, *hydra*'s mail router will send the letter directly to *csl.sri.com*. On first blush, this seems to be the best option; however, it has drawbacks. Not every host can transmit all types of mail; for example, *icarus* is the only host that can be used to send *uucp* mail, because it is the only host with telephone lines connected. If some distant site changes its internet address, all machines must have their host tables updated (since some machines, such as *hydra*, do not use domain name resolving); worse, when a configuration file is changed to reflect a change in the way mail is handled, all configuration files must be changed. This leads to problems of mail being caught in transit when the change occurs.

Since all nonlocal mail must be sent through the gateway machine, why not send all nonlocal mail there for forwarding? In this configuration, whenever any letter being mailed has an address on some other machine, it is sent directly to *icarus*. *Icarus* then decides how to send it to its destination. For example, if a letter is sent to *csl.sri.com* from *hydra*, *hydra*'s mail router will send the letter to *icarus*; then *icarus*'s mail router will send the letter on to *csl.sri.com*. This scheme has advantages over the previous one, in that all hosts except the gateway need only know which host is the gateway; if there is any mail on the host for any other machine, it is sent to the gateway, which then decides how best to deliver it. If some distant site changes its internet address, only the host table on the gateway need be changed; the tables on the other hosts may be updated as convenient, but doing so (or failing to do so) will not affect the transfer of mail. One would suspect that this configuration is worse than the previous one, because if *icarus* goes down, no mail can be sent off site. However, since all our network traffic goes through *icarus*, if *icarus* is down, no off-site mail could be sent

†VAX is a Trademark of Digital Equipment Corporation.

anyway. The only problem is that intrasite mail will not go from one machine to another when *icarus* is down.

There is one other advantage to the latter configuration. RIACS uses the domain naming system of the ARPANET. As a result, having a central machine provides a very easy way to decide which machine is to be listed as "riacs.edu". All other machines are invisible to off site systems; any letter going out, regardless of the machine from which it originated, has the return address changed to read "*user*@riacs.edu". In essence, we can reconfigure our machines, change their internet addresses, and so forth without having to have people throughout the country update their host tables. Only the address of the machine answering to "riacs.edu" need be kept the same.

For these reasons, RIACS uses the latter configuration. Pictorially, this system is arranged as shown in Figure 2.

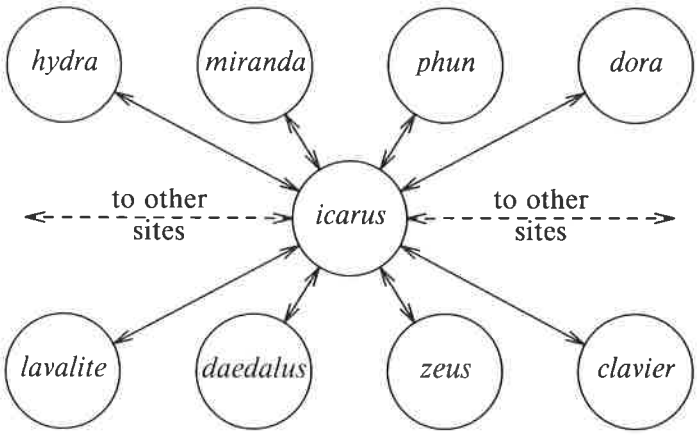


Figure 2. The RIACS Mail Configuration

There are two different configuration files: the first, the *gateway* version, resides on the RIACS gateway and handles routing over a variety of communications media; the second, the *non-gateway* version, resides on all other hosts.

3. *sendmail* Configuration File Preliminaries

This section describes options, macros, classes, mailers, and other characteristics of the configuration files. Rulesets are described in the next section.

3.1. Macros

Certain macros and options are set in both the gateway and non-gateway configuration files. Some are necessary for *sendmail* to work; others are peculiar to RIACS. In this section, those common settings are explained.

The first three macros in the gateway's file set the domain name **N**, the uucp host name **U**, and the version **Z** of this *sendmail* configuration file; in the non-gateway file, the uucp host name is not given but the gateway host **G** is. *sendmail* also requires six macros to be defined; otherwise, it will not function properly. Table 2 summarizes the definitions of these macros.

Table 2. RIACS Settings for Macros		
macro	gateway setting	non-gateway setting
G	*not set*	icarus.riacs.edu
N	riacs.edu	
U	riacs.uucp	*not set*
Z	2.0G	2.0N
j	$w	
e	$j Sendmail $v/$Z ready at $b	
l	From $g $d	
n	MAILER-DAEMON	
o	.:%@!^=/[]	
q	$?x$x $.<$g>	

3.2. Options

The configuration files also set many options for *sendmail;* both the gateway and non-gateway configuration files set the same ones. Table 3 summarizes their settings.

Table 3. RIACS Configuration File Options			
option	RIACS setting	option	RIACS setting
A	/usr/lib/aliases	**a**	10
B	_	**d**	b
F	0600	**g**	1
H	/usr/lib/sendmail.hf	**o**	*set*
L	9	**r**	1h
Q	/usr/spool/mqueue	**s**	*set*
S	/usr/lib/sendmail.st	**t**	PST,PDT
T	3d	**u**	1
W	*	**x**	8
X	12		

Some of these options require a bit of explanation.

A,a These refer to the alias file feature of *sendmail*. The alias file is located in */usr/lib/aliases*, and if *sendmail* detects the alias file is being rebuilt it will wait up to ten minutes before deciding the rebuild has failed and initiate one of its own.

B Older mailers allowed the use of blank characters in addresses; this causes all sorts of problems on many systems. This option directs *sendmail* to replace all blanks with underscores '_'. RIACS used to use periods '.', but this poses problems in a domain system; an underscore does not.

d This option instructs the daemon to run in background mode; the daemon will disconnect from the terminal.

F,Q The values of these options are the mode and location of the temporary and queue files used by *sendmail*. It is a RIACS policy that the mode shall be at least mode 0660, and preferably mode 0600; these settings prevent others from reading or altering mail while it is in the queue. On the gateway, this has never posed a problem; but it has on the Suns running NFS, since *root* on a client Sun cannot write to the server's queue directory. Worse, even if it could, the client's *sendmail* process could not read the files it put there! At this point, there are two options: either make the queue files mode 0660 and the queue directory 0777 (meaning anyone can delete queue files, although not necessarily read them), or make each Sun's queue directory a private one (by linking the queue directory to a directory in */private*). At RIACS, the latter is done; on the server, */usr/spool/mqueue* is a symbolic link to */private/usr/spool/mqueue*.

g,u When *sendmail* spawns a subprocess to deliver mail, it resets the user and group identification numbers to 1 (for *daemon*). This provides a measure of security; if the mailer were running as *root*, a breach of security could result in compromise of the entire system rather than just daemons.

L All mail coming into and going out of RIACS is logged; the sender and recipient are named in the log file, as is the queue number and other useful information. To reduce the amount of logging, lower this number. Logging is done via *syslog*(3), so to determine where information is being sent, look there.

o Since many mailers use spaces rather than commas to delimit names, this option instructs *sendmail* to accept and handle such lists.

r This option instructs *sendmail* to time out after 1 hour during a connection to send mail. Supposedly, this should "never happen," but we all know that the real world rarely lives up to our expectations. With the current state of the internet, a connection is often broken but one or both ends do not know this. Better to time out than wait forever (or for the next reboot).

S This is where statistics on *sendmail* are kept. The file does not grow, so the */usr* file system will not overflow due to this.

s This is a safety measure; it ensures the queue is always current, even when *sendmail* will deliver the message immediately. With mail, paranoia is a healthy state of mind!

T After 3 days, the message will be returned as undeliverable. Another popular value is 15 days.

t This is a relic of V6 UNIX and is ignored here.

W This option applies to versions of *sendmail* compiled with the "wizard" option (4.3 BSD) or the "debug" option (4.2 BSD). Those versions of *sendmail* provide a special set of commands, ostensibly for debugging, that allow anyone with access to the SMTP server to break into the system. **DO NOT DELETE THIS OPTION!** With this option set, an attacker cannot break in this way even if the running version of *sendmail* contains the security hole.

X,x These control how load averages affect SMTP connections. If the load average is 8 or more, incoming messages are queued and not delivered. If the load average is 12 or more, all requests for an SMTP connection are refused.

3.3. Precedences

This section is ignored unless one of the header fields is a "Precedence:" field, in which case the message is given the appropriate precedence with respect to all other undelivered messages. By default all messages have precedence 0 (the same as "first-class"). Table 7 summarizes the precedences RIACS recognizes.

Table 7. RIACS Mail Precedence Levels

class	priority
junk	-100
first-class	0
special-delivery	100

3.4. Trusted Users

When forwarding mail, *sendmail* must sometimes specify a different sender than in the message headers (this occurs in intra-network mailing). These users can use the appropriate command flags to do this; no-one else can. Putting a user in this field means he can change the sender. Currently, the trusted users are *root*, *daemon*, *uucp*, and *network*.

3.5. Mailers

This section describes the mailers for the gateway. There are five of them: *local*, which delivers mail with recipients on that host, *prog*, which delivers mail to programs or servers on that host, *uucp*, which delivers mail to the uucp network, *tcp*, which delivers mail over the internet, and *utcp*, which delivers uucp mail over the internet. Non-gateway hosts use three mailers (*local*, *prog*, and *tcp*) with identical descriptions to the mailers for the gateway.

The mailers each have flags indicating what actions *sendmail* should take when invoking them; these all follow the "F=" in each entry. All mailers require "Date: ", "From: ", and "Message-Id: " fields (flags **D**, **F**, and **M**) so if any of those fields are missing *sendmail* will add them before the mailer is invoked. The *local* and *prog* mailers both perform final delivery (the flag **l**); the letter will not be passed to another delivery agent when one of these mailers is invoked. Both the *local* and *tcp* mailers can deliver letters to multiple addresses simultaneously (the flag **m**) so *sendmail* will issue only one command to send the letter, rather than one such command per addressee. UNIX mailers do not conform to the standard, usually requiring headers of the form "From *user* ..." to be the first line in the letter; since the *local* mailer adds those lines, *sendmail* will not (the flag **n**); also, *sendmail* is to specify the recipient with the **-r** argument to the *local* mailer (the **r** flag). All mailers but the *tcp* mailer require quotes to be stripped from the address before being called, and *sendmail* does this (the **s** flag). The *prog*, *tcp*, and *utcp* mailers are expensive to connect to, so the gateway will only connect during a queue run; notice the **e** flag. (This flag only has an effect if the "c" flag is specified in the Options section. RIACS does not do this.) Also, the *uucp* mailer requires a special line at the top of the letter (this line is of the form "remote from *host* ..." and is also a violation of the standard) so the **U** flag has *sendmail* add it before sending the letter to that mailer. The *uucp*, *tcp*, and *utcp* mailers preserve the case of letters in user names (the **u** flag), and the *uucp* and *utcp* mailers also preserve the case of letters in host names (the **h** flag). Finally, the *tcp* and *utcp* mailers have a line length limit as specified in RFC 821 [RFC821]; *sendmail* will split lines in the letter if need be to keep lines short enough (the **L** flag).

Three mailers have other special considerations. Due to limits inherent in the *uucp* network, it is exceedingly unwise to send a letter with more than $2^{16}-1$ characters; the field "M=65535" in the *uucp* and *utcp* mailers prevents *sendmail* from sending files larger than that through this mailer. The *tcp* and *utcp* mailers use the two-character string "<CR><LF>" (that is, a carriage return followed by a line feed) to signal the end of a line.

The *uucp* mailer is peculiar to 4.3 UNIX. When called, */usr/bin/uux* is invoked as "uux - -z -a$f -gA $h!rmail ($u)"; this just means *uux* will read a letter from the standard input, send error messages to the sender (the **$f** is replaced with the sender's address), queue the message in the UUCP mail system with high priority (the "-gA" does this), and sends the input to the command *rmail* on the remote host **$h** with the recipient's name in parentheses on the command line.

The *tcp* mailer is really a "pseudo-mailer," because *sendmail* is the delivery agent. The string "P=[IPC]" tells *sendmail* to use an SMTP protocol to transmit the message. No other program is invoked (there is no program named "IPC", in fact). By default, communication is done over port 25. If some other port should be used, name the port after the "$h" in the "A=" string; for example, "A=IPC $h 100" initiates contact over port 100.

The *utcp* mailer is a compromise between the *uucp* mailer and the *tcp* mailer. Some sites on the internet are also uucp hosts, so rather than queue the message within the uucp system and send it that way, we use the SMTP protocol to transmit it directly. Thus, the mailer has the characteristics of both the *uucp* and *tcp* mailers in its description, but uses the *uucp* mailer's rewriting rules to rewrite addresses.

3.6. Headers

The header lines indicate what header fields should be added. The "Received:" field is inserted into every message, and the others are inserted depending on what flags the mailers in the *mailers* section above have. Table 5 shows which of the other headers are inserted into letters being handled by the mailers, and the flag present in the "F=" field of the mailer description that causes the header line to be inserted.

Table 5. Table of Header Lines

header line	*inserted for these mailers*	*flag*
Return-Path:	*none*	**P**
Resent-Date:	local, prog, uucp, tcp	**D**
Resent-Message-Id:	local, prog, uucp, tcp	**M**
Message-Id:	local, prog, uucp, tcp	**M**
Date:	local, prog, uucp, tcp	**D**
Resent-From:	local, prog, uucp, tcp	**F**
From:	local, prog, uucp, tcp	**F**
Full-Name:	*none*	**x**

If any of these lines are in the header, another new one will **not** be added. The test is done on the flag and not the header itself. For example, if a message has the header field "Date:", neither a new "Date:" nor a "Resent-Date" header field will be added, since a field tied to the **D** flag is present.

The next two sections describe the rulesets used to rewrite and deliver mail.

4. *sendmail* Gateway Configuration File

When *sendmail* obtains an address for processing, it runs that address through a series of rules grouped into rulesets. These rulesets can change the address, pass it on unchanged, or substitute a new address entirely for it. This section describes what RIACS' configuration file does.

4.1. Domain Names and Canonization

All the rulesets, except ruleset 3, assume the address is in a standard form; this reduces the number of rules that each ruleset needs. The form assumed is that the domain to which the letter is to be sent is surrounded by angle brackets '<', '>'. For example, the *canonical* form of "mab@riacs.edu" is "mab<@riacs.edu>"; the canonical form of "@purdue.edu:mab@riacs.edu" is "<@purdue.edu>:mab@riacs.edu" because the letter is to be sent to "purdue.edu", and from there to "riacs.edu". This form is internal; the angle brackets are removed before the address is written back into the header, or passed to the mailer. It is simply a convenience that allows simpler rewriting rules.

There are some cases where no legitimate domains exist to cover the addresses; for example, UUCP is not a domain in the sense that RFC 920 [RFC920] defines domain. But it is much easier to pretend that there is a domain ".uucp", and deal with addresses in that context; we can use the rewriting rules for legitimate domains to process addresses for these sites too. So the address "megatest!mab" is canonize to "mab<@megatest.uucp>". Also, many people use UUCP addressing to route letters through Internet sites; such addresses have the form "decwrl.dec.com!megatest!mab". In these cases, the canonization process does *not* add a ".uucp" to the domain; it uses the given domain. So, the above address would be canonized as "megatest!mab<@decwrl.dec.com>". This approach is actually a bit perilous, because many sites use domain names even though they are not on the Internet; such cases are handled by specific rules affecting only the address handed to a transport mechanism. This way, we can use the same canonical form of an address for all cases.

4.1.1. Ruleset 3

Ruleset 3 canonizes all addresses. We shall go through this ruleset in detail, showing how each rule moves the address towards canonization. When *sendmail* gets an address, it may be surrounded with '<', '>'; so we must strip these off. First, we handle the case where no address is provided, then we get the innermost address:

```
R<>                     $@@
R$*<$*<$*<$+>$*>$*>$*   $4
R$*<$*<$+>$*>$*         $3
R$*<$+>$*               $2
```

The first rule simply replaces the empty address with an "@" sign, and returns that as the new address. Other rulesets will deal with it later on. The next two rules deal with multiple nestings of addresses. This is not covered by the standard, so it is ambiguous. The convention most places seem to hav adopted is to treat the innermost pair of angle brackets as delimiting the address. The final rule simply treats whatever is in angle brackets as the address.

The next rule,

```
R$+ at $+               $1@$2
```

simply rewrites an archaic address specification to an acceptable one (RFC 733 [RFC733] allowed it, but RFC 822 [RFC822] does not). The old form of the address is now illegal, but since many sites will be slow to convert, it still should be recognized.

The next rule converts route specifications into an internal form:

```
R@$+,$+                 @$1:$2
```

This form replaces all ','s with ':'s, so (for example) "*@site1,@site2:user@site3*" would become "*@site1:@site2:user@site3*". This is not a legal form, but dealing with this form rather than the legal one makes rulesets much simpler. (The address is rewritten to the legal form before being output.) Note that we wish to forward the message to the first host in the specification.

Next, we deal with some of the more obvious errors.

```
R:$+                    $@$>3$1
R@:$*                   $@$>3$1
R$*@                    $@$>3$1
```

These three rules handle the case of a null domain name; in the first case, the separator ':' is deleted, and in the last two cases, the '@' introducing the null domain name is stripped and the address reprocessed.

Now we are ready to canonize:

```
R@$+:$+                 $@<@$1>:$2
R$+@$+                  $:$1<@$2>
R$+<$+@$+>              $1$2<@$3>
R$*<@$*.>$*             $@$>3$1@$2$3
R$+<@$+>                $@$1<@$2>
```

The first rule handles route-specified addresses; note that it returns the result, since later rules shift the angle brackets right and mail is sent to the leftmost host in the route specified addresses rather than the rightmost. The second rule adds angle brackets to other addresses, and the third rule ensures that they are placed around the rightmost domain name. The fourth line deals with an error situation: when a domain name ending in a '.' is given, it is an error, so the offending '.' is deleted and the name recanonized. Otherwise, the result of the canonization is returned.

The last section of this ruleset handles syntaxes of addresses not falling within the scope of [RFC822]. These include UUCP and BITNET (among others).

```
R$+^$+                  $1!$2
R$-!$+                  $@$2<@$1.uucp>
```

```
R$+!$+          $@$2<@$1>
R$-:$+          $@$2<@$1>
R$-.$+          $@$2<@$1>
R$-=$+          $@$2<@$1.bitnet>
R$+%$+          $@$>9$1%$2
```

The first three rules deal with UUCP. The first rule converts an old UUCP address to the new form, and the second and third rules do the conversions described at the beginning of this section. The following two rules convert BERKNET addressing syntax to that of [RFC822]. The rule after than deals with BITNET syntax; like UUCP, BITNET is a pseudo-domain ".bitnet" and is dealt with similarly. The last rule translates "%" to "@" when appropriate; this is done by invoking ruleset 9, which will be discussed next.

4.1.2. Ruleset 9

One very common situation arising on the Internet is that of network routing; for example, if someone at an ARPANET site wishes to send a letter to someone at a CSNET site, he must indicate that the message is to go to another network, the CSNET network. This requires the message to be sent to a site on both ARPANET and CSNET (such sites are called *relay sites* or *relays*). The obvious syntax is to use routing, as "@relay.cs.net:postmaster@vpi.csnet", but a far more common syntax is to write "postmaster%vpi.csnet@relay.cs.net". It is more desirable to have the mail routing mechanism worry about how to get the message from ARPANET to CSNET. With sites that allow this (such as RIACS), people tend to write "postmaster%vpi.csnet", that is, use a "%" rather than a "@". The proper way to deal with such an address is to send the message to the *last* site named. Ruleset 9 deals with addresses involving sequences of "%"; it changes the final such character to "@" and canonizes the result:

```
R$*%$*          $1@$2
R$*@$*@$*       $1%$2@$3
R$*@$*          $@$1<@$2>
```

The first rule changes all "%" characters in the address to "@" characters and the second rule changes all but the last "@" back. The third rule adds the '<' and '>' around the last site. This canonizes the name. Note that if no domain or site name is given after the "%", ruleset 9 is not called, so no error checking need be done. In fact, throughout the other syntaxes ruleset 3 recognizes, there is an implicit assumption that no attempt will be made to recover from a faulty address. Such addresses will cause the mail to fail and be returned to the sender at some point, either at RIACS or at a site farther along the mail path.

4.2. The Transport Mechanism

Ruleset 0 is the basis for transport. By the time this ruleset returns, the input address must be resolved to a mailer, a host, and an address. When ruleset 0 returns, *sendmail* instructs the named mailer to contact the named host and send the letter to the named address. In most cases, the mailer is determined by the domain, although there is a mechanism that allows the name of the site to determine the transport mechanism. By default, the pseudo-domain ".uucp" uses the *uucp* mailer; and other domains either use the *tcp* mailer or are rewritten to send to a relay site. The relay site may use either the *tcp* or *uucp* mailers (currently, they all use the *tcp* mailer). All addresses given to the *tcp* mailer are enclosed in angle brackets and contain the name of the host to which the letter is being given, because many hosts reject addresses not in angle brackets or without the name of the host.

4.2.1. Ruleset 0

The first few rules of ruleset 0 rewrite the address into a form that can be analyzed and turned into a mailer/host/address set. The first rule eliminates loops back to RIACS; this is an efficiency consideration, and could be dropped.

```
R$*<@$+>$*      $:$>6$1<@$2>$3
```

We shall discuss how this is done when we look at ruleset 6, below. The next rule handles messages with no addresses.

```
R@                      $#local$:$n
```

Recall that ruleset 3 changed null addresses into "@"; this rule just sends the letter to the designated person. The next rule deals with domains to which RIACS does not have direct access:

```
R$*<@$+>$*              $:$>8$1<@$2>$3
```

These must be reached through relays; all this resolution is done in ruleset 8, which is used to rewrite addresses here. Note that some subdomains of known domains are rewritten; this need not be done once RIACS uses a name resolver, but should be kept just in case the name resolver is replaced. The rewriting must be done if the top-level domain is one RIACS can only access through a relay (such as BITNET or CSNET).

Finally we must convert route specification addresses from the internal form to the legal form. This rule does so:

```
R$*:@$*                 $1,@$2
```

Wherever the sequence ":@" appears, it is replaced by ",@". This converts the internal format of a route specified address to the legal form, which can then be output.

The address is now in a form that can be resolved to a mailer/host/address triple. Some RIACS hosts do not accept SMTP connections, and others are not accessible via *uucp*; table 6 summarizes the mailers used to reach each local host. They are divided into two classes based on the mailer used.

Table 6. Mailers for RIACS Local Hosts

host	mailer	class	host	mailer	class
clavier	tcp	**T**	icarus	tcp	**T**
cube	uucp	**U**	lavalite	tcp	**T**
daedalus	tcp	**T**	miranda	tcp	**T**
dora	tcp	**T**	phun	tcp	**T**
hydra	tcp	**T**	zeus	tcp	**T**

Rather than being rigid and rejecting all incorrectly-routed mail, mail is rerouted properly. The next four rules do this:

```
R$*<@$=T.uucp>$*        $#tcp$@$2$:<$1@$2$3>
R$*<@$=U>$*             $#uucp$@$2$:$1$3
R$*<@$=U.$N>$*          $#uucp$@$2$:$1$3
R$*<@$=U.arpa>$*        $#uucp$@$2$:$1$3
```

The first rule changes any mail sent to a RIACS SMTP host over *uucp* to use the SMTP mailer *tcp*. The next three rules change any mail sent to a RIACS host that does not accept mail from the *tcp* mailer to use *uucp*.

One last problem remains. Many sites in the "uucp" (and other) domains use the ARPA domains, and so mail must be routed to them on a per-site basis. Also, many "uucp" sites use the ARPANET as the communications medium for *uucp*; it makes more sense to send mail over the ARPANET rather than use *uucp*. This ruleset is used to do such routing. No mailers are invoked; the addresses are simply rewritten. Table 7 describes the sites the addresses of which are rewritten:

Table 7. Addresses that RIACS Rewrites			
site names		mailers	
incoming	outgoing	original	replaced by
amdcad.amd.com	amdcad	*tcp*	*uucp*
ames	ames.arc.nasa.gov	*uucp*	*utcp*
decwrl	decwrl.dec.com	*uucp*	*utcp*
ll-xn	ll-xn.arpa	*uucp*	*utcp*
lll-crg	lll-crg.arpa	*uucp*	*utcp*
rutgers	rutgers.rutgers.edu	*uucp*	*utcp*
seismo	seismo.css.gov	*uucp*	*utcp*

The rules are:

```
R$*<@amdcad.amd.com>$*$#uucp$@amdcad$:$1$2
R$*<@ames.uucp>$*      $#utcp$@ames.arc.nasa.gov$:<$1@ames.arc.nasa.gov$2>
R$*<@decwrl.uucp>$*    $#utcp$@decwrl.dec.com$:<$1@decwrl.dec.com$2>
R$*<@ll-xn.uucp>$*     $#utcp$@$[ll-xn$]$:<$1@$[ll-xn$]$2>
R$*<@lll-crg.uucp>$*   $#utcp$@$[lll-crg$]$:<$1@$[lll-crg$]$2>
R$*<@rutgers.uucp>$*   $#utcp$@rutgers.rutgers.edu$:<$1@rutgers.rutgers.edu$2>
R$*<@seismo.uucp>$*    $#utcp$@seismo.css.gov$:<$1@seismo.css.gov$2>
```

More rules can be added as needed.

We next dispose of addresses within the "uucp" pseudo-domain:

```
R<@$+.uucp>:$+         $#uucp$@$1$:$2
R$+<@$+.uucp>          $#uucp$@$2$:$1
```

Note that these handle "uucp" addresses in both route specifications and other forms. Two rules are necessary because the name of the machine to which the letter is being sent is not included in the address; for example, a letter to "megatest!ametek!root" would call the *uucp* mailer with a host of "megatest" and an address of "ametek!root".

The next rule handles all addresses for the *tcp* mailer:

```
R$*<@$*>$*             $#tcp$@$2$:<$1@$2$3>
```

Note that the address handed to the mailer is enclosed in angle brackets. This handles both rout-specified addresses and other forms of addressing. Finally, any address not converted to a mailer/host/address triple is intended to be delivered on this machine, so the final rule in ruleset 0 does this:

```
R$+                    $#local$:$1
```

4.2.2. Ruleset 6

Earlier we mentioned some other rulesets; let us now look at ruleset 6. This ruleset eliminates the gateway as a target for a letter; otherwise, the gateway would just deliver mail to itself, a rather pointless exercise especially when it can be handled immediately. The rules in this ruleset are actually divided into two groups: the ones that do canonization, and the ones that recurse. The first six rules all recanonize after stripping the gateway's name; the next six do not change the address, but reinvoke this ruleset if the target host is the gateway.

The first two rules delete the top-level domain and uucp names from the address:

```
R$*<@$N>$*             $>3$1$2
R$*<@$U>$*             $>3$1$2
```

Note these simply delete the host name in angle brackets and recanonize. The next four deal with the name of the gateway:

```
R$*<@$=w>$*            $>3$1$3
```

```
R$*<@$=G>$*          $>3$1$3
R$*<@$=G.$N>$*       $>3$1$3
R$*<@$=G.arpa>$*     $>3$1$3
```

Because the gateway is known by so many different names, and the domain may or may not be appended, the first two rules deal with the gateway host name without the domain, the third with the fully qualified gateway host name, and the last with the old ".arpa" form of the gateway host name. The next six rules are similar, but just reinvoke this ruleset:

```
R$*<@$N>$*           $>3$1<@$N>$2
R$*<@$U>$*           $>3$1<@$U>$2
R$*<@$=w>$*          $>3$1<@$2>$3
R$*<@$=G>$*          $>3$1<@$2>$3
R$*<@$=G.$N>$*       $>3$1<@$2.$N>$3
R$*<@$=G.arpa>$*     $>3$1<@$2.arpa>$3
```

They are present in case the recanonization has focused on another version of the gateway's name.

4.2.3. Ruleset 8

Ruleset 8 deals with the domains to which RIACS does not have direct access. This requires the address to be rewritten in order to send the message through a relay host. There are two syntaxes used. The first, for unrouted addresses, is to replace the address with something the relay can use. For example, the address "root@munnari.oz" would become "root%munnari.oz@seismo.css.gov" since "seismo.css.gov" is the relay host for the domain "oz". The relevant rules are:

```
R$*<@decwrl.dec.com>$*$@$1<@decwrl.dec.com>$2
R$*<@decwrl.dec>$*   $@$1<@decwrl.dec.com>$2
R$*<@$+.au>          $@$1%$2.au<@seismo.css.gov>
R$*<@$+.bitnet>      $@$1%$2.bitnet<@wiscvm.wisc.edu>
R$*<@$+.csnet>       $@$1%$2.csnet<@relay.cs.net>
R$*<@$+.dec.com>     $@$1%$2.dec<@decwrl.dec.com>
R$*<@$+.dec>         $@$1%$2.dec<@decwrl.dec.com>
R$*<@$+.decnet>      $@$1%$2.decnet<@$[ames-io$]>
R$*<@$+.mailnet>     $@$1%$2.mailnet<@$[mit-multics$]>
R$*<@$+.oz>          $@$1%$2.oz<@seismo.css.gov>
R$*<@telemail>       $@$1%telemail<@orion.arc.nasa.gov>
```

The first two rules make sure that letters being sent to "*someone*@decwrl.dec.com", the DEC Easynet relay, are not addressed to "*someone*%decwrl.dec@decwrl.dec.com" (for some reason, "decwrl.dec.com" cannot handle this). The remaining nine rules simply rewrite the addresses as required. Be aware that the rules for "dec.com" will go away when the name resolver software becomes reliable; in fact, since RIACS has the table of DEC hosts, all rules involving "dec" (except "decnet", which is an Ames local network) may soon be commented out.

The second syntax is used for route-specified addresses; the relevant rules are:

```
R<@$+.au>$*          $@<@seismo.css.gov>:$1.au:$2
R<@$+.bitnet>$*      $@<@wiscvm.wisc.edu>:@$1.bitnet:$2
R<@$+.csnet>$*       $@<@relay.cs.net>:@$1.csnet:$2
R<@$+.dec.com>$*     $@<@decwrl.dec.com>:@$1.dec:$2
R<@$+.dec>$*         $@<@decwrl.dec.com>:@$1.dec:$2
R<@$+.decnet>$*      $@<@$[ames-io$]>:@$1.decnet:$2
R<@$+.mailnet>$*     $@<@$[mit-multics$]>:@$1.mailnet:$2
R<@$+.oz>$*          $@<@seismo.css.gov>:$1.oz:$2
R<@telemail>$*       $@<@orion.arc.nasa.gov>:@telemail:$2
```

The rules just put the relay before the host to which the letter is to be directed; for example, the address "<@munnari.oz>:*someone@otherhost*" will be rewritten as

"<@seismo.css.gov>:@munnari.oz:*someone@otherhost*". Notice that addresses to "decwrl.dec.com" are taken care of by the first two rules in the ruleset, and so need not be repeated here.

4.3. The Sender's Address

There are two philosophies for handling a sender's address. Either the address is rewritten to reflect the way the message is sent, or it is not rewritten unless failing to do so would produce known errors when that path is used as a reply path. RIACS takes the latter approach. We assume that other sites know best how to route their mail, and so (with the exception of *uucp*, for a reason explained below) we do not rewrite the sender's address unless the sender is at this site.

The rewriting rulesets applied to the sender's address depends in part on the mailers being used. In all cases, ruleset 1 is applied, then the mailer-dependent rules, and finally ruleset 4 operates on the result.

4.3.1. Ruleset 1

Ruleset 1 hides the name of the local host within RIACS by replacing it with the name of the gateway as a domain; so, "root@hydra" would be rewritten as "root@riacs.edu". This enables RIACS to hide its internal configuration from the rest of the world. It is most useful should the internal configuration change, since only the gateway would need to be updated. Table 8 summarizes this configuration; note the nicknames do not have any domain or pseudo-domain names appended.

Table 8. RIACS Local Hosts

official host name	nicknames
clavier.riacs.edu	clavier riacs-clavier
cube.riacs.edu	cube hypercube riacs-cube riacs-hypercube
daedalus.riacs.edu	daedalus riacs-daedalus ames-daedalus
dora.riacs.edu	dora riacs-dora
hydra.riacs.edu	hydra riacs-hydra
icarus.riacs.edu	icarus riacs riacs-icarus riacs-gw
lavalite.riacs.edu	lavalite riacs-lavalite
miranda.riacs.edu	miranda riacs-miranda
pegasus.riacs.edu	pegasus riacs-pegasus ames-pegasus
phun.riacs.edu	phun riacs-phun
zeus.riacs.edu	zeus riacs-zeus

This ruleset is quite simple, consisting of only four rules:

```
R$*<@$H>$*          $@$1<@$N>$3
R$*<@$=H.arpa>$*    $@$1<@$N>$3
R$*<@$=H.$N>$*      $@$1<@$N>$3
R$*<@$=H.uucp>$*    $@$1<@$U>$3
```

The first rule changes unqualified names, the second changes old-style ".arpa" names, and the third changes fully qualified hostnames. Notice the fourth, which translates local host names into the standard RIACS *uucp* designator.

4.3.2. Ruleset 10

The *local* and *prog* mailers next use ruleset 10, which consists of one rule:

```
R@                  $n
```

If there is a null address, this ensures the appropriate person gets the (rejected) letter; that person can then decide what action to take.

4.3.3. Ruleset 13

The *uucp* mailer uses ruleset 13. This ruleset must prepend the RIACS uucp name *riacs.uucp* to the address, so for example, "mab@<megatest.uucp>" would become "megatest!mab<@riacs.uucp>". If this is not done, the remote *uucp* mail handler will record the return address as "somewhere!mab" rather than "riacs!mab", making it impossible to reply to that letter. Five rules accomplish this:

```
R$*<@$U>$*          $@$1<@$U>$2
R$*<@$+.uucp>       $@$2!$1<@$U>
R<@$+.uucp>$*       $@<@$U>:@$1.uucp:$2
R$*<@$*>$*          $@$1<@$2>$3
R$*                 $@$1<@$U>
```

The first rule simply eliminates cases where *riacs.uucp* is already in the address. The next two rulesets add the name to addresses sent via *uucp* and via routing specifications. The fourth rule returns any nonlocal addresses; presumably, these are of the form "*someone*@amdcad.amd.com" where RIACS can only talk to "amdcad.amd.com" via *uucp*. In this case, the remote site's *uucp* mail agent must be able to handle the address; RIACS should not have to. The final rule adds the uucp domain to addresses from a local machine, so "mab" would be rewritten as "mab<@riacs.uucp>".

4.3.4. Ruleset 14

The *tcp* mailer uses ruleset 14. If the sender is local, this ruleset appends the RIACS domain to the address:

```
R$*<@$*>$*          $@$1<@$2>$3
R$+                 $:$1<@$N>
```

4.3.5. Ruleset 4

Finally, ruleset 4 does some miscellaneous cleanup:

```
R@                  $@
```

Just in case an empty address gets this far, it is ignored; the next mailer can deal with it. (This should never happen, but just in case this line is left here.) The next rule decanonizes the address by deleting the angle brackets '<', '>':

```
R$*<$+>$*           $1$2$3
```

Route-specified addresses must be completely surrounded by such brackets, and converted to a legal form. The next rules do this:

```
R$+:@$+             $1,@$2
R@$+:$+@$+          <@$1:$2@$3>
```

and the final rule rewrites uucp addresses to their usual form:

```
R$*@$+.uucp         $2!$1
```

Note that this will not affect uucp addresses which are part of a route specification.

4.4. The Recipient's Address

As with senders' addresses, there are two philosophies for handling a recipient's address. Either the recipient's address is not rewritten unless failing to do so would produce known errors when that path is used by the next host to which the message is sent, or the address is rewritten to reflect the routing used to get to the next host in the address. RIACS takes the former approach. We assume that other sites know best how to route their mail, and so we do not rewrite the recipient's address unless the recipient is at this site.

The rewriting rulesets applied to the recipient's address depends in part on the mailers being used. In all cases, ruleset 2 is applied, then the mailer-dependent rules, and finally ruleset 4 operates on the result.

4.4.1. Ruleset 2

sendmail runs the address given to the mailer through these rulesets, so ruleset 2 is used to recanonize these addresses. Note that this also works for recipient addresses handed directly to ruleset 2 (that is, those from the "To:" or "cc:" field) because defocusing and recanonizing is essentially a no-op in this case. Ruleset two has two rules:

```
R$*<$*@$*>$*        $:$1$2$3$3
R$*                 $@$>3$1
```

The first rule deletes all angle brackets, and the second rule just invokes ruleset 3.

4.4.2. Ruleset 20

The *local* and *prog* mailers next use ruleset 20, which needs to do nothing, so there are no rules in it.

4.4.3. Ruleset 23

The *uucp* mailer uses ruleset 23, which (like ruleset 20) needs to do nothing, so there are no rules in it.

4.4.4. Ruleset 24

The *tcp* mailer uses ruleset 24. If the recipient is local, this ruleset appends the RIACS domain to the address:

```
R$*<@$*>$*          $@$>8$1<@$2>$3
R$+                 $:$1<@$N>
```

Ruleset 4 has been discussed above.

4.5. Conclusion

This completes the discussion of why the gateway configuration file was set up as it is. Now let us look at the non-gateway file.

5. *sendmail* Non-Gateway Configuration File

This file is much simpler, because the strategy is simply to send any nonlocal mail to the gateway. Accordingly, sender and recipient addresses are not processed, and canonization is done only to aid in determining the mailer to be used.

5.1. The Transport Mechanism

Rulesets 3 and 9 are the same as for the gateway configuration file, so they will not be discussed again, and ruleset 0 is much simpler.

5.2. Ruleset 0

As with the gateway, the first rule in that ruleset ensures that all names of the host are deleted by invoking ruleset 6:

```
R$*                 $:$>6$1
```

At this point, the mailer can be determined. Either the letter is local, in which case the *local* or *prog* mailers need to be invoked, or it is not local, in which case the *tcp* mailer sends the letter to the gateway:

```
R@                  $#local$:$n
R$*<$*@$*>$*        $#tcp$@$G$:<$1$2@$3$4>
R$+                 $#local$:$1
```

Note that if the address is empty, the first rule will send the letter to the appropriate user, who can take whatever action is deemed appropriate. The next rule just forwards any mail with a host name to the gateway. Since names for this host were deleted by the first rule in this ruleset, such mail is destined for another host. The last rule just delivers the mail locally.

5.2.1. Ruleset 6

Ruleset 6 is somewhat different because it does not have to deal with domain names or "uucp" names, but only the names of this machine. As in the gateway configuration file, the first set of rules recanonize the address, and the next set determines whether or not to reinvoke ruleset 6:

```
R$*<@$=w>$*          $>3$1$3
R$*<@$=w.arpa>$*     $>3$1$3
R$*<@$=w.$N>$*       $>3$1$3
R$*<@$=w>$*          $>6$1<@$2>$3
R$*<@$=w.arpa>$*     $>6$1<@$2.arpa>$3
R$*<@$=w.$N>$*       $>6$1<@$2.$N>$3
```

5.3. Philosophy

The philosophy in writing the non-gateway configuration files has been to keep the rules as simple as possible. This is why local hosts will send mail to each other through the gateway rather than directly; since not all hosts can speak SMTP, only the gateway needs to keep track of how each host is to receive mail. This makes the gateway file the only one that needs to be changed when the configuration changes.

6. Debugging a *sendmail* Configuration File:

RIACS runs various versions of *sendmail* on its hosts. All are essentially the same program, so the techniques discussed below work for all versions of *sendmail*. In all cases, it is **strongly** recommended that the new configuration file **not** be installed until it is fully tested and debugged; instead, run *sendmail* with the **-C** option. So, to debug a configuration file *new.cf*, issue the command

`sendmail -Cnew.cf` *other options ...*

The first mode for testing is called *test mode*, appropriately enough, and is used to run addresses through rulesets to show what each ruleset is given, what it returns, and how the given address is rewritten. To use it, put the flag **-bt** on the command line. Here is a sample session, using the gateway configuration file; some of the longer lines have been split into two, in order to fit on the page.

```
% /usr/lib/sendmail -bt
ADDRESS TEST MODE
Enter <ruleset> <address>
> 0 riacs.edu!site.com!xyz
rewrite: ruleset  3   input: "riacs" "." "edu" "!" "site" "." "csnet" "!" "xyz"
rewrite: ruleset  3 returns: "site" "." "csnet" "!" "xyz" "<" "@" "riacs" "."
                             "edu" ">"
rewrite: ruleset  0   input: "site" "." "csnet" "!" "xyz" "<" "@" "riacs" "."
                             "edu" ">"
rewrite: ruleset  6   input: "site" "." "csnet" "!" "xyz" "<" "@" "riacs" "."
                             "edu" ">"
rewrite: ruleset  3   input: "site" "." "csnet" "!" "xyz"
rewrite: ruleset  3 returns: "xyz" "<" "@" "site" "." "csnet" ">"
rewrite: ruleset  6 returns: "xyz" "<" "@" "site" "." "csnet" ">"
rewrite: ruleset  8   input: "xyz" "<" "@" "site" "." "csnet" ">"
rewrite: ruleset  8 returns: "xyz" "%" "site" "." "csnet" "<" "@" "relay" "."
                             "cs" "." net" ">"
```

```
rewrite: ruleset  0 returns: "$#" "tcp" "$@" "relay" "." "cs" "." "net" "$:"
                             "<" "xyz" "%" "site" "." "csnet" "@" "relay" "."
                             "cs" "." "net" ">"
> 1,14,4 cde@phun.riacs.edu
rewrite: ruleset  3   input: "cde" "@" "phun" "." "riacs" "." "edu"
rewrite: ruleset  3 returns: "cde" "<" "@" "phun" "." "riacs" "." "edu" ">"
rewrite: ruleset  1   input: "cde" "<" "@" "phun" "." "riacs" "." "edu" ">"
rewrite: ruleset  1 returns: "cde" "<" "@" "riacs" "." "edu" ">"
rewrite: ruleset 14   input: "cde" "<" "@" "riacs" "." "edu" ">"
rewrite: ruleset  8   input: "cde" "<" "@" "riacs" "." "edu" ">"
rewrite: ruleset  8 returns: "cde" "<" "@" "riacs" "." "edu" ">"
rewrite: ruleset 14 returns: "cde" "<" "@" "riacs" "." "edu" ">"
rewrite: ruleset  4   input: "cde" "<" "@" "riacs" "." "edu" ">"
rewrite: ruleset  4 returns: "cde" "@" "riacs" "." "edu"
> 2,24,4 ghi@vpi.csnet
rewrite: ruleset  3   input: "ghi" "@" "site" "." "au"
rewrite: ruleset  3 returns: "ghi" "<" "@" "site" "." "au" ">"
rewrite: ruleset  2   input: "ghi" "<" "@" "site" "." "au" ">"
rewrite: ruleset  3   input: "ghi" "@" "site" "." "au"
rewrite: ruleset  3 returns: "ghi" "<" "@" "site" "." "au" ">"
rewrite: ruleset  2 returns: "ghi" "<" "@" "site" "." "au" ">"
rewrite: ruleset 24   input: "ghi" "<" "@" "site" "." "au" ">"
rewrite: ruleset  8   input: "ghi" "<" "@" "site" "." "au" ">"
rewrite: ruleset  8 returns: "ghi" "%" "site" "." "au" "<" "@" "seismo" "."
                             "css" "." "gov" ">"
rewrite: ruleset 24 returns: "ghi" "%" "site" "." "au" "<" "@" "seismo" "."
                             "css" "." "gov" ">"
rewrite: ruleset  4   input: "ghi" "%" "site" "." "au" "<" "@" "seismo" "."
                             "css" "." "gov" ">"
rewrite: ruleset  4 returns: "ghi" "%" "site" "." "au" "@" "seismo" "." "css"
                             "." "gov"
```

In this example, the first ruleset called is ruleset 3 (this is always the case, regardless of what rules you name). It is passed the line "riacs.edu!site.csnet!xyz", and returns the canonized form "site.csnet!xyz<@riacs.uucp>". Ruleset 0 is then invoked; its first rule calls ruleset 6, which strips off the "@riacs.edu" (since this example was run on *icarus*, which is also known as *riacs.edu*, there is no point whatsoever in remailing the message to *riacs.edu*) and calls ruleset 3 on the remainder of the string, "site.csnet!xyz". Ruleset 3 rewrites this as "xyz<@site.csnet>" and returns to ruleset 6, which returns control to ruleset 0. Ruleset 0 then calls ruleset 8, which rewrites the address to send it through the appropriate relay ("xyz%site.csnet<@relay.cs.net>") and then returns this new form to ruleset 0. Ruleset 0 rewrites the address as "$#tcp$@relay.cs.net$:<xyz%site.csnet@relay.cs.net>" and using this form calls the appropriate mail handling agent. The other two examples work in the same way, the second example using rulesets 3, 1, 14, and 4, and the third example using rulesets 3, 2, 24, and 4.

The version of *sendmail* on *icarus*, the RIACS gateway, has been modified in two ways. The first, relevant to debugging, prints control strings used in the configuration file as they are entered in that file [GILL86]. This modification has not been made to any other version of *sendmail*, so beware! Usually, the final line of the first case in the above example would be printed as:

```
rewrite: ruleset  0 returns: "^V" "tcp" "^W" "site" "." "com" "^X" "<" "xyz"
     "@" "site" "." "com" ">"
```

for 4.3 BSD's version of *sendmail*, or

```
rewrite: ruleset  0 returns: "^U" "tcp" "^V" "site" "." "com" "^W" "<" "xyz"
     "@" "site" "." "com" ">"
```

for other versions of *sendmail*. Table 9 shows the internal symbols corresponding to the special symbols used in rewriting rules; the version of *sendmail* on *icarus* uses those in the 4.3 BSD column, and all other machines use those in the 4.2 BSD column.

Table 9. Table of Configuration File Actions

Symbol	*Shown* 4.2BSD	4.3BSD	*Symbol*	*Shown* 4.2BSD	4.3BSD	*Symbol*	*Shown* 4.2BSD	4.3BSD
$*	^P	^P	$	^T	^U	$?	^Y(?)	^Z
$+	^Q	^Q	$#	^U	^V	$|	^Z(?)	^[
$-	^R	^R	$@	^V	^W	$.	^[(?)	^\
$=	^S	^S	$:	^W	^X	$[	*none*	^]
$~	^\	^T	$>	^X	^Y	$]	*none*	^^

There is also a set of debugging flags that are quite useful when one particular aspect of *sendmail* needs to be examined very closely. They differ from version to version; the ones listed in Table 10 are for version 5.51.

Table 10. Table of Debugging Flags

flag	*level*	*information about*
0	1	main: run as daemon in foreground
0	4	main: show canonical name and aliases of current host
0	15	main: print configuration table as loaded from configuration file
0	44	main: print command-line arguments to program
1	1	main: report sender of letter
1	9	main: allow user doing debugging to substitute new sender
2	1	exit: report exit status, flags
5	4	event: report alarms
5	5	event: report setting, clearing of event
5	6	event: show event processing
6	1	error: report how mail is dealt with on error
6	5	error: display state of *sendmail* on error
7	1	queueing: show names of queue file
7	2	queueing: show name of temporary queue file ("tf" file)
7	20	queueing: report processing of the queued file
10	1	deliver: show address given to mailer
11	1	deliver: show address to which letter is actually sent
12	1	parseaddr: show resolution of remote name in address
13	1	deliver: show to whom messages are to be sent
13	3	deliver: check for errors in sending messages
13	4	deliver: report to whom errors go
14	2	header: print list of names in header with commas
15	1	daemon (server): report requests
15	2	daemon (server): report forking to process a request
15	15	daemon (server): put network in debugging mode
16	1	daemon (client): report making remote connection
16	14	daemon (client): put network in debugging mode
18	1	smtp: report result of trying to make connection
18	100	smtp: pause after error in reading from remote connection
20	1	parseaddr: report address to be parsed
21	2	parseaddr: report input, output of each ruleset
21	3	parseaddr: report call to another ruleset (via "$>")
21	4	parseaddr: report result of call to another ruleset
21	10	parseaddr: report failure of ruleset to match

Table 10. Table of Debugging Flags

flag	*level*	*information about*
21	12	parseaddr: report ruleset to be matched and if match succeeds
21	15	parseaddr: report substitution due to matches substitution
21	35	parseaddr: show attempts to match
22	36	parseaddr: show address prescan
22	45	parseaddr: show what is being prescanned
22	101	parseaddr: show states during parsing of address
25	1	recipient: show recipient list (multiple addresses possible)
26	1	recipient: show an individual address
27	1	alias: report alias definition
30	1	smtp: report end of headers
30	2	smtp: report addition of "Apparently-To:" field
30	3	smtp: report processing of UNIX "From" line
31	6	header: show header line being read
32	1	header: show all headers to current letter
33	1	header: show header line as processed
35	9	macro: report definition
35	24	macro: show macro expansion
36	5	symbols: report addition and/or lookup
36	9	symbols: display symbol hash
37	1	configuration: show option settings in configuration file
40	1	queue: show queue run of file
40	4	queue: print queue file lines ("qf" file) as read
41	2	queue: report problem reading queue during ordering
45	1	envelope: report sender
50	1	envelope: report deallocation
51	4	queue: show deletion of transcript file
52	1	main: report disconnection from controlling terminal
52	5	main: do not disconnect from controlling terminal

Options are set with the **-d**flag, which has the format **-d***flaglist.level*. To set flags 5, 6, 7, and 36 at level 10, for example, give the option "-d5-7,36.10". It is usually advisable to confine debugging flags to number 21 (which deals with the way addresses are parsed) or display all of them; also, specifying a level prints all messages from lower levels, too. To print any information that could be of importance, use "-d0-99.99"; the levels higher than 99 are used to debug *sendmail*'s internals.

As stated above, debugging flag 21 is useful enough to merit some special discussion. When more detail about the address parsing is needed, this flag prints each step in the ruleset and indicates how a match is being made. For example,

```
% /usr/lib/sendmail -bt -d21.99
Version 5.51
ADDRESS TEST MODE
Enter <ruleset> <address>
> 0 mab@riacs.edu
rewrite: ruleset  3   input: "mab" "@" "riacs" "." "edu"
-----trying rule: "<" ">"
ap="mab", rp="<"
----- rule fails
-----trying rule: "$*" "<" "$*" "<" "$*" "<" "$+" ">" "$*" ">" "$*" ">" "$*"
ap="mab", rp="$*"
ap="mab", rp="<"
ap="@", rp="<"
```

```
ap="riacs", rp="<"
ap=".", rp="<"
ap="edu", rp="<"
ap=<null>, rp="<"
----- rule fails
-----trying rule: "$+" "@" "$+"
ap="mab", rp="$+"
ap="@", rp="@"
ap="riacs", rp="$+"
ap=".", rp=<null>
ap="edu", rp=<null>
-----rule matches: "$:" "$1" "<" "@" "$2" ">"
$1: 7fffe2bc="mab"
$2: 7fffe2c2="riacs" 7fffe2c8="." 7fffe2ca="edu"
rewritten as: "mab" "<" "@" "riacs" "." "edu" ">"
rewrite: ruleset  3 returns: "mab" "<" "@" "riacs" "." "edu" ">"
rewrite: ruleset  0   input: "mab" "<" "@" "riacs" "." "edu" ">"
-----trying rule: "$*" "<" "@" "$+" ">"
ap="mab", rp="$*"
ap="mab", rp="<"
ap="<", rp="<"
ap="@", rp="@"
ap="riacs", rp="$+"
ap=".", rp=">"
ap="edu", rp=">"
ap=">", rp=">"
-----rule matches: "$:" "$>" "6" "$1" "<" "@" "$2" ">"
$1: 7fffe2bc="mab"
$2: 7fffe2c2="riacs" 7fffe2c8="." 7fffe2ca="edu"
-----callsubr 6
rewrite: ruleset  6   input: "mab" "<" "@" "riacs" "." "edu" ">"
-----trying rule: "$+" "<" "@" "riacs" "." "edu" ">"
ap="mab", rp="$+"
ap="<", rp="<"
ap="@", rp="@"
ap="riacs", rp="riacs"
ap=".", rp="."
ap="edu", rp="edu"
ap=">", rp=">"
-----rule matches: "$>" "3" "$1"
$1: 7fffe2bc="mab"
-----callsubr 3
rewrite: ruleset  3   input: "mab"
rewrite: ruleset  3 returns: "mab"
rewritten as: "mab"
rewrite: ruleset  6 returns: "mab"
rewritten as: "mab"
-----trying rule: "$+"
ap="mab", rp="$+"
-----rule matches: "$#" "local" "$:" "$1"
$1: 7fffe2bc="mab"
rewritten as: "$#" "local" "$:" "mab"
rewrite: ruleset  0 returns: "$#" "local" "$:" "mab"
>
```

(Note that parts of the above were edited to keep the output brief.) When ruleset 3 is first called, it tests to see if the address matches "<>". It does not, as the token "mab" is not the same as the token '<'. So it tries the next rule in the ruleset, and continues, until it finds a match in the rule "$+@$+". This rule replaces the previous form with "mab<@riacs.edu>", and continues the ruleset. Finally, it returns with "mab<@riacs.edu>". Ruleset 0 then calls ruleset 6, which strips off the "<@riacs.edu>" and calls ruleset 3 to recanonize the rest of the address "mab". Eventually, ruleset 6 returns "mab", and ruleset 0 turns this into the appropriate mailer command.

Occasionally a user will report a problem with mail requiring the maintainer of the mail system to determine at which hosts, if any, the problem occurred. To determine this, look at the headers in the letter that posed the problem. If there are error messages, read them first; if those do not provide enough information, scan the "Received: " lines. Lines added at RIACS indicate from whom the mail routers think the message was received; this can be used to track backwards. As an example, the message with the headers

```
Received: from icarus.riacs.edu (icarus.ARPA) by hydra.riacs.edu (4.12/1.6N)
            id AA14571; Fri, 16 Jan 87 16:41:02 pst
Received: from RELAY.CS.NET by icarus.riacs.edu (5.51/1.6G)
            id AA00679; Fri, 16 Jan 87 16:40:49 PST
Received: from icarus.riacs.edu by RELAY.CS.NET id aa02271; 16 Jan 87 19:30 EST
Received: by icarus.riacs.edu (5.51/1.6G)
            id AA00630; Fri, 16 Jan 87 16:29:16 PST
Message-Id: <8701170029.AA00630@icarus.riacs.edu>
Date: Fri, 16 Jan 87 16:29:16 PST
From: Matt Bishop <mab@riacs.edu>
To: mab%riacs.edu@RELAY.CS.NET
Subject: show how "Received: " header lines work
```

went from *icarus* (*icarus.riacs.edu*) to *RELAY.CS.NET* and back to *icarus* and from there to *hydra*.

Incidentally, the version of *sendmail* on *icarus* has been modified to report the name of the system from which *icarus* receives the letter. This modification defines the unsupported macro **s** so that the name of the sending host may be included on the "Received" header line, as well as allowing the receiving host to recognize nicknames as well as official names (these two fixes are from [LOVS86]). The *sendmail* files which have been altered are stored in their original form as *file*.orig in the *sendmail* source directory.

7. Installing New *sendmail* Configuration Files

In general, to install a new *sendmail* configuration file, do the following:

1. Locate and kill any currently-running invocations of *sendmail*. This is best done by typing "ps gaux | grep sendmail" as *root* and then killing all the processes this command lists. If some have died anyway, do not worry; they exited between the "ps" and the "kill".
2. Copy */usr/lib/sendmail.cf* somewhere. This way, if the new configuration file does not work right, the old one can be put back.
3. Move the new configuration file to */usr/lib/sendmail.cf*.
4. Freeze this file. This puts it into a form *sendmail* can load quickly. Type "/usr/lib/sendmail -bz"; the file that is created is */usr/lib/sendmail.fc*.
5. Restart the *sendmail* daemon. This command will be of the form "/usr/lib/sendmail -bd -q1h", but it varies from system to system. Look in */etc/rc.local* for the exact command; typing "grep 'sendmail.*-bd' /etc/rc.local" will print the exact command.
6. Type "mailq". On the server Sun, the command will print a message saying "Freeze file out of date", and delete the frozen configuration file; this is normal. (Why it is done is not known.) This ensures that the *sendmail* program is working correctly.

On the client Suns, the procedure is slightly different since they use the server's configuration file. On these systems, omit steps 2 through 4.

8. Changing the Configuration Files

This section describes how to make routine changes to the *sendmail* configuration files. It does not cover massive rewriting; for that, your best bet is to find a *sendmail* guru or just experiment.

8.1. Adding a New RIACS Host (Gateway Configuration File)

This is really quite easy. First, figure out how mail will be sent to the host; if it will go via *uucp* or *tcp*, you're in luck. (If not, you'll have to define a new mailer. See [ALLM84b] for a description of how to do this; use the already-created mailer descriptions as a guide.)

First, add all names of the new host in the class **H**. If the new host were named "hera.riacs.edu", with nicknames "riacs-hera.arpa", "hera", "ames-hera", and "riacs-hera", add a line of the form

CHhera riacs-hera ames-hera

to the appropriate section (where the rest of the hosts are defined). Then, if the mailer is *tcp*, add all the names to the class **T**; if the mailer is *uucp*, add all the names to the class **U**. If you are using some other mailer, define a new class for the mailer (remember, *sendmail* macro names are one character only!), and put this host into the class. For explanatory purposes, call the new class **W** and the mailer *newmail*. Then in ruleset 0, add the following rules to the end of the "resolve the mailers ... handle local hosts" section:

```
R$*<$=W>$*          $#newmail$@$2$:address
R$*<$=W.arpa>$*     $#newmail$@$2$:address
R$*<$=W.uucp>$*     $#newmail$@$2$:address
R$*<$=W.$N>$*       $#newmail$@$2$:address
```

(where address is the form of the address newmailer expects). Then reinstall the configuration file.

8.2. Sending UUCP Mail to A New Host Over the Internet (Gateway Configuration File)

These changes allow you to route mail to internet hosts over the internet, even when the sender has asked that mail be sent via *uucp*. It is strongly recommended that this be done only when *uucp* uses the internet as the communications medium; this ensures the site is really on the internet. Needless to say, this will not work unless the site accepts SMTP connections.

Suppose RIACS talks to the site "faraway.sub.dom" (with *uucp* name *faraway*) using *uucp* running over the internet. To route mail through SMTP but in such a way everyone else thinks the mail was sent through the uucp system, add the line

```
R$*<@faraway.uucp>$*  $#utcp$@faraway.sub.dom$:<$1@faraway.sub.dom$2>
```

to ruleset 0, in the section "...handle special hosts". It can go anywhere in that section.

See the section entitled **Mailers** for a description of the *utcp* mailer.

8.3. Sending Internet Mail to A New Host Over UUCP (Gateway Configuration File)

These changes allow you to route mail to uucp hosts which use the internet domain naming scheme, even when the sender has asked that mail be sent via *uucp*. This simply must be done; it should be undone when (and if) the site joins the internet (in which case perhaps mail routed through *uucp* should be sent via the internet, as described in the section above).

Suppose RIACS talks to the site "faraway.sub.dom" (with *uucp* name *faraway*) using *uucp* running over a telephone line. It is not possible to pretend the message was sent over the internet, due to the limits of the uucp mailers, but adding the line

```
R$*<@faraway.sub.dom>$*$#uucp$@faraway$:$1$2
```

to ruleset 0, in the section "...handle special hosts" will route such mail over *uucp*. (This rule can go anywhere in that section.) Be aware the address will be run through the *uucp* mailer's sender and recipient address rewriting rules, so the receiving host should get a path to which it can reply. The emphasis is on the word "should."

8.4. Adding a New Domain Relay Site (Gateway Configuration File)

Suppose RIACS requires access to the domain "newdom", but that domain can only be reached through the host "relay.dom.net". The goal is to accept addresses like "*user@site*.newdom" and route the mail automatically.

Ruleset 8 is used to handle these cases. Two lines must be added, one for route addressing, and the other for unrouted addressing. In the section of this ruleset entitled "output fake domain stuff in user%host.fake@relay-host syntax", add a rule of the form

```
R$*<@$+.newdom>        $@$1%$2.newdom<@relay.dom.net>
```

(which maps "*user*<@*site*.newdom>" to "*user%site*.newdom<@relay.dom.net>") and to the section of this ruleset entitled "output fake domain stuff in route-specific syntax", add a rule of the form

```
R<@$+.newdom>:$*       $@<@relay.dom.net>:@$1.newdom:$2
```

(which maps "<@*site*.newdom>:*user@site2*" to "<@relay.dom.net>:@*site*.newdom:*user@site2*"). Then addresses without the relay site named explicitly will be sent to the relay site anyway.

8.5. Change the Gateway Host (Gateway and Non-Gateway Configuration Files)

To do this, both the gateway and non-gateway files must be changed. In the gateway file, change the definition of the class **G** to be all the names of the new gateway host. In the non-gateway file, change the definition of the macro **G** to be the name of the new gateway host (it is recommended this be the fully qualified internet name). Then install the gateway configuration file on the new gateway host, and the non-gateway configuration file on all other hosts.

This assumes the gateway is to handle all outgoing traffic. If not, a third configuration file must be written for the UUCP host. See the introductory comments to this section, above. (Translation: good luck!)

9. Conclusion

The goal of this work has been to make the RIACS mail system into a low maintenance system and, when maintenance is necessary, easy to maintain. This document is an integral part of that plan, because it is intended to allow a systems programmer to understand how the mail configuration files work, and why the mail system is set up as it is. Needless to say, only time will tell if these rather ambitious goals have been met.

Acknowledgement

My thanks to Barry Leiner for his comments on an earlier draft.

References

[ALLM84a] Allman, Eric, "Sendmail – An Internetwork Mail Router", in *UNIX System Manager's Manual, 4.2 Berkeley Software Distribution, Virtual VAX-11 Version*, Computer Science Division, Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA (Mar. 1984), as reprinted by the USENIX Association

[ALLM84b] Allman, Eric, "Sendmail – Installation and Operation Guide Version 4.2", in *UNIX System Manager's Manual, 4.2 Berkeley Software Distribution, Virtual VAX-11 Version*, Computer Science Division, Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA (Mar. 1984), as reprinted by the USENIX Association

[GILL86] Gilliam, Jeff, USENET message <2821@voder.uucp>, National Semiconductor, Santa Clara, CA (Dec. 11, 1986).

[LOVS86] Lovstrand, Lennart, USENET message <377@majestix.liuida.uucp>, Computer Science Department, University of Linkoping, Sweden (Dec. 28, 1986).

[RFC733] Crocker, D. H., Vittal, J. J., Pogran, K. T., and Henderson, D. A., *Standard for the Format of ARPA Network Text Message*, ARPANET Request for Comments No. 733, Network Information Center, SRI International, Menlo Park, CA (Nov. 1977)

[RFC821] Postel, Jonathan B., *Simple Mail Transfer Protocol*, ARPANET Request for Comments No. 821, Network Information Center SRI International, Menlo Park, CA (Aug. 1982)

[RFC822] Crocker, David H., *Standard for the Format of ARPA Network Text Message* (revised), ARPANET Request for Comments No. 822, Network Information Center SRI International, Menlo Park, CA (Aug. 1982)

[RFC920] Postel, J., and Reynolds, J., *Domain Requirements*, ARPANET Request for Comments No. 920, Network Information Center SRI International, Menlo Park, CA (October 1984)

[ROSE86] Rose, Marshall T., and Romine, John L., *The RAND MH Message Handling System: User's Manual*, UCI Version (Apr. 1986)

[UPM84] –, *UNIX User's Manual Reference Guide, 4.2 Berkeley Software Distribution, Virtual VAX-11 Version*, Computer Science Division, Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA (Mar. 1984), as reprinted by the USENIX Association

Nominations for the 1988 Election of the USENIX Board of Directors

Lew Law has been requested by Alan Nemeth, President of the Board, to assemble and chair a Nominating Committee to propose a slate of candidates for election to the Board of Directors in 1988.

This is pursuant to Article 7 of the Association's bylaws, which require notification of the membership of the names of those serving on the Nominating Committee at least nine months prior to the elections.

The following have agreed to serve on the Committee:

Lew Law (chair)	{usenix,harvard}!law 617-495-2627
Bruce Borden	hplabs!dana!bsb 408-732-0400
Tom Ferrin	tef@cgl.ucsf.edu 415-476-1100
Mike O'Dell	mo@seismo 703-893-3660
Peter Langston	psl@bellcore 206-641-6106
Mike Tilson	mike@hcrvax 416-922-8397

The Board is comprised of eight members: President, Vice President, Treasurer, Secretary, and four Board members.

Some attributes which contribute to promoting the smooth functioning and continuing development of the Association are:

- a real interest in what the USENIX Association is doing
- a committment to improve and expand the Association's activities
- willingness and available time to commit to Association activities

If you are interested, know of anyone who is or might be interested, or know of anyone who would be an asset to the USENIX Association as a Board member, and willing to serve, please contact Lew Law or any member of the Nominating Committee.

The committee would like and appreciate as much help as it can get!

≡ ≡ ≡

Ten Years ago in UNIX NEWS

Here is the fix to the *ttyn*(III) problem mentioned in the February issue of *UNIX News*. . .

After the line that reads:

```
mov   buf+2,(sp)
```

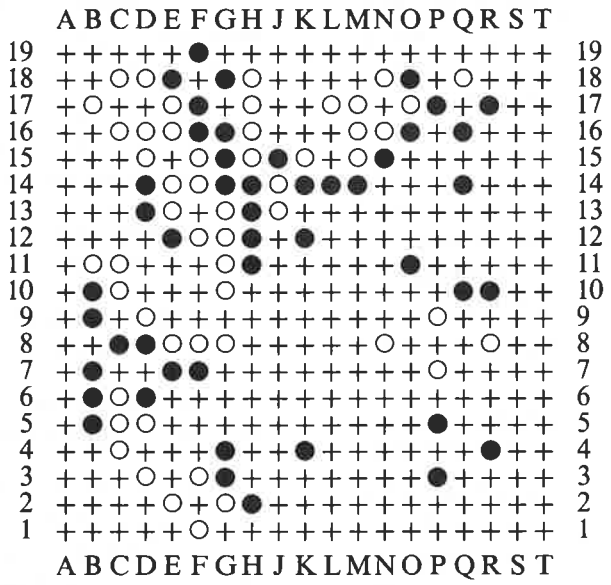
I inserted:

```
mov   buf,r1
sys   stat;dev;buf
bes   er1
cmp   buf,r1
bne   er1
```

A similar change has to be made to *nroff*(I), file: *s7/nroff1.s*. This file contains a slightly different version of *ttyn*.

George Rolf
Catholic University
Nijmegen

The Fourth Annual USENIX Computer Go Tournament and Championship



Announcement

The fourth annual USENIX Computer Go Tournament will be held on Wednesday, June 10th, during the Summer 1987 USENIX conference in Phoenix, Arizona.

All interested parties are invited to submit programs. Programs may be written in any language as long as the binary will run under a UNIX operating system like 4.? BSD. The tournament rules will be essentially those established for the first USENIX Computer Go Tournament (see below).

This event will be a "championship." The winner will be the "USENIX Computer Go Champion" until the next championship is held (probably at the USENIX Conference the following summer).

Conference attendees may bring programs to submit with them as long as they get in touch with Peter Langston NO LATER THAN noon on the day before the tournament (preferably earlier). He can be reached through the USENIX Conference Office. People who are unable to attend the conference but would like to enter their programs can do so by sending a compilable source to one of the addresses below, (or by taking a chance and sending an "executable" file which may, or may not, function under last minute operating systems changes or machine changes, or ...).

The source code for the referee program to be used has been distributed through netnews "net.sources" and can be redistributed if interest warrants.

Comments or programs can be sent via electronic mail to *bellcore!psl* or *psl@BELLCORE.ARPA*. U.S. Mail should be sent to:

Peter Langston
Bell Communications Research
MRE 2D-396
435 South Street
Morristown, NJ 07960

USENIX Computer Go Tournament Rules

Revised April, 1986
P. Langston

- **A full size board will be used.** The board will be 19×19 with columns labeled "A" through "T" (excluding "I") left to right, and rows labeled "19" through "1" top to bottom.
- **Komi will be 5.5 points.** The second player gets a 5.5 point bonus.
- **There will be a time limit.** Each program will be limited to a total of 60 minutes of

accumulated "user" time. If a program goes over the time limit it will only be allowed 10 seconds of "user" time for each move (byo-romi). Subsequently, if a program uses more than 10 seconds of "user" time for a move it will immediately forfeit the game.

• **The programs must not be idle unnecessarily.** If 10 minutes of "real" time elapse with no increase in the current program's "user" time, it will be assumed that the program is stuck and the program will forfeit. (This rule is included to handle cases where a program loses synchronization or is doing something like: `for (;;) read(0, buf, sizeof buf);`.)

• **There will be no forking (around).** Each program must be a single process and must not fork other processes. Forking interferes with the timing mechanism and, like any attempt to evade or fool the timing, will result in a forfeit. The tournament will use a "referee" program to execute each competing pair of programs. There will be no command-line arguments, i.e. argc will be 1. All communication with the programs will be via the standard input and standard output, thus the programs must understand a specific set of commands and generate output of a specific form.

a) All input commands to the competing programs will be in the form of lines of text appearing on the standard input and terminated by a newline.

b) The first line of input to each program will be either "black" or "white" (lower case) to indicate which color the program will be playing (and thereby whether the program plays first or second).

c) The placement of a stone will be expressed as upper-case letter-number (e.g. "G7" note capitalization).

d) A pass will be expressed as "pass" (lower case).

e) The command "byo-romi" (lower case) means the time limit has been exceeded and all further moves must be generated within the 10 second time limit.

f) All output from the competing programs will be in the form of lines of characters sent to the "standard output" terminated by a newline, and had better either be flushed after every line or be unbuffered to start with (e.g. `setbuf(stdout, 0);`).

g) Any output lines not beginning with "A" through "T" (excluding "I") or "pass" will be considered garbage and ignored.

Any syntactically correct but semantically illegal move will be considered a forfeit. The three possibilities are: playing on a non-empty spot (occupied or off the board), ko violation, and suicide.

• **Play will end when both programs pass in sequence.** The programs may pass at any time, but once both pass concurrently, the game is over.

• **The decisions of the judge will be final.** A human judge will evaluate each game's results and may fill in missed dame or may judge a game incomplete if, in the judge's opinion, too much is unresolved. In general, Japanese rules will be used, (Nihon Kiin).

≡ ≡ ≡

We've Moved!

At the end of March, the USENIX Association moved to new office space. Our new postal address is:

P.O. Box 2299
Berkeley, CA 94710

The telephone number remains 415-528-8649.

Report on the Large Installation System Administrators' Workshop

Seventy people attended the first Large Installation System Administrators' Workshop in Philadelphia, PA, April 9-10, 1987. Each attendee was asked to present a one or two page paper detailing a certain aspect of system administration at their installation. The papers were then sorted into the following categories: network administration, backup and restore, software distribution and synchronization, mail and news, accounts and accounting, reliability enhancements, and I/O management. In addition to the papers, panel discussions and/or question and answer periods followed each of the sessions. Rob Kolstad, CONVEX Computer Corporation, and a member of the USENIX Association Board of Directors, and Alix Vasilatos, MIT Project Athena co-chaired the workshop. Their behind-the-scenes planning and on-site management of the sessions insured the success of the program. A questionnaire was distributed to the participants and their reflections on the two days indicated that the workshop was an outstanding success. It brought together a group of people who have a focused interest and allowed them an opportunity to share their problems/successes. This, and the ability to interact with other attendees in a one-on-one basis, were the most satisfying aspects of the workshop. Several attendees suggested that another workshop be considered a year from now in order to follow up on the progress generated by this workshop. A summary of the sessions follows.

Network Administration

Since all of the participants represented large installations, administering a network was a common concern. Security and convenience were the main topics in this session. Security seemed to be more of a concern to commercial companies than the university attendees. These companies often have third party developers using their systems and they need to protect areas such as research and development from outside access. Procedures ranged from simple encryption to elaborate, and expensive, call back features on dial-in ports. Even in-house departments, such as personnel, need to take precautions to safeguard files. It was generally felt that the read, write, execute permissions standard to most systems don't offer the needed protection. Convenience issues in a networked environment centered on transparency and reliability. A user may want to use a laser writer attached to one of the systems on the network, but doesn't want to go through some convoluted sequence of commands to access the device. Backup and recovery are also features that should be kept transparent to the user. Invaluable is the system administrator who restores the "lost" file/directory while the user continues along his productive path. Reliability can be achieved through redundancy (expensive) or maintaining a common environment across all machines on the network. When a machine is going to be out of commission for a day or two, the affected user(s) can be moved to another machine in a matter of an hour or so.

Backup and Restore

The range of services and resources varied widely among the installations represented. Some sites only had enough disk space for a day or two of incremental dumps, while others kept as much as three weeks of information on-line. Once the information needed to be dumped to tape, 6250 drives with 3600 foot tapes were often used. As much as 225 megabytes of data can be stored on this medium. Most attendees indicated that after three years of use the tapes needed to be relegated to the circular file, while some installations are still using tapes that are ten years old. Interestingly enough, some installations did incremental dumps on active systems. Production environments, where down time to do incremental dumps is not possible, performed the backups while the systems were in use. The time slot assigned to this task was often in the middle of the night when use of the system was reduced, limiting the number of files which might be affected. Restoration of files/directories often required the system administrator's intervention because of the need for super user privileges.

Some sites had enough faith in their operations staff to allow them to have limited superuser privileges to carry out the recovery. One site charged a $25 fee, in hard cash, every time a restore operation was requested. The money was required up front, and a significant drop in requests was experienced when the policy was instituted.

Software Distribution and Synchronization

The ideal environment is one in which all hardware/software is identical across the network. Software distribution can be automated from a single server and installation insured to maintain commonality among all systems. Although such environments exist, we all know they are not the rule. The other extreme is to put out an update and let the various system administrators decide what they want to implement. Yes, a few of these installations were represented, as well. Most seemed to agree that a central source for software distribution is desirable and more efficient.

Mail and News

One company represented had over 5,000 machines worldwide they wanted to make sure could receive mail from the home office. Demanding that each machine have a common mail system enabled this to happen. Having a news facility to address questions/answers from the user community was implemented at another installation. As entries were aged off the system they were placed on a passive news system. Each time a submission was made, the entire data base was searched to determine if it had been addressed before. Reliability of mail systems was another issue. Can you be sure the mail reached its destination? Just because it didn't come bouncing back to you doesn't mean it didn't fall into a sinkhole. Responding to mail is one way to insure it was received, but do you respond to the response so that they respondee will know you received his response...? News has become the bane of some administrators' existences, while it is an inalienable right to others. The amount of resources expended on shipping various news groups around the world is staggering.

Accounts and Accounting

University environments constantly need to prune the password file of users who have become inactive. However, inactive users often surface at a later date and want their accounts re-activated. Automating a procedure to add, delete and re-activate users eases this task. Charge back accounting in a university often includes a variety of algorithms for students, staff, faculty, and even commercial accounts. Accounting files are often used to identify abuses in a system and correct them before they become too great of a problem. Students often find it a challenge to develop ways to "beat the system."

Reliability Enhancements

How does one insure the reliability of computer resources? One way is through redundancy of hardware, but this is expensive. Military applications and air traffic control operations are often forced to go this expensive route. An environment that consists of similar machines running the same software provides a fairly reliable situation. Moving users from system to system is simple and can be implemented when a certain machine is out of service for a lengthy period of time.

I/O Management

Remote device usage often requires that you know the path to the machine hosting that device. A presentation in this session described a network where each machine knew only the devices that were available locally, and passed unknown requests to a parent system in search of the device. The request was passed up the tree until it found the resource it needed, or arrived at the root node where the location of all devices in the network was kept. To expedite the request, a *.destination* file in the user's login directory could be used to identify the path to the remote device. Routing multiplexors were also discussed in this session. Users are able to connect to a variety of systems in the network through this method. The pros and cons of various multiplexor manufacturers' offerings were also discussed.

Summary

The true success of a meeting is gauged by feedback from the participants. Over 65% of the attendees took the time to fill out the questionnaire and the results were overwhelmingly positive. Administrators with common problems/solutions were brought together and the interaction afforded by a workshop of limited size was one of the most-often mentioned advantages they appreciated. Most of the solutions that were presented are in the public domain and available from the presenters. A proceedings consisting of the papers is available on request, as long as they last, from Rob Kolstad, CONVEX Computer Corporation, 701 Plano Road, Richardson, TX 75081.

John Donnelly

≡ ≡ ≡

Future Meetings

4th Computer Graphics Workshop Oct. 8 & 9, 1987, Cambridge, MA

Abstracts are due by May 1, 1987. For information, contact Tom Duff, the program chair, at *research!td* or (201) 582-6485.

USENIX 1988 Winter Conference and UniForum – Dallas

The USENIX 1988 Winter Conference will be held on February 10-12, 1988, at the Registry Hotel in Dallas, Texas. It will be concurrent with UniForum 1988, which will also be in Dallas. The Conference will feature tutorials and technical sessions.

USENIX 1988 Summer Conference and Exhibition – San Francisco

The USENIX 1988 Summer Conference and Exhibition will be held on June 21-24, 1988, at the Hilton Hotel in San Francisco, California. There will be a conference, tutorials, and vendor exhibits.

Long-term USENIX Conference Schedule

Jun 8-12 '87 Hyatt Regency, Phoenix, AZ
Feb 10-12 '88 Registry Hotel, Dallas, TX
Jun 21-24 '88 Hilton Hotel, San Francisco, CA
Feb 1- 3 '89 Town & Country Inn, San Diego, CA
Jun 13-16 '89 Hyatt Regency, Baltimore, MD
Feb '90 Washington, DC
Jun 11-15 '90 Marriott Hotel, Anaheim, CA
Jun '91 Nashville, TN

≡ ≡ ≡

Prize Winning Papers

The following are the award winners for the best student papers submitted to the Phoenix Program Committee:

Bob Gray, University of Colorado: Automatic Error Recovery in a Fast Parser

Irving Reid, University of Saskatchewan: RPCC – A Stub Compiler for SUN RPC

Publications Available

The following publications are available from the Association Office or the source indicated. Prices and overseas postage charges are per copy. California residents please add applicable sales tax. Payments **must** be enclosed with the order and **must** be in US dollars payable on a US bank.

USENIX Conference and Workshop Proceedings

Note the reduction in price of all USENIX conference proceedings more than a year old.

Meeting	Location	Date	Price	Overseas Mail Air	Overseas Mail Surface	Source
USENIX	Wash. DC	Winter '87	$20	$25	$5	USENIX
UniForum	Wash. DC	'87	$20	$20		/usr/group
Graphics Workshop III	Monterey	December '86	$10	$15	$5	USENIX
USENIX	Atlanta	Summer '86	$10	$25	$5	USENIX
USENIX	Denver	Winter '86	$10	$25	$5	USENIX
UniForum	Anaheim	'86	$15	$20		/usr/group
Graphics Workshop II	Monterey	December '85	$ 3	$ 7	$5	USENIX
USENIX	Dallas	Winter '85	$10	$25	$5	USENIX
Graphics Workshop I	Monterey	December '84	$ 3	$ 7	$5	USENIX
USENIX	Salt Lake	Summer '84	$10	$25	$5	USENIX
UniForum	Wash. DC	'84	$ 5	$20		/usr/group

USENIX Association
P.O. Box 2299
Berkeley, CA 94710

/usr/group
4655 Old Ironsides Dr., #200
Santa Clara, CA 95050

All t-shirts will be on sale in Phoenix at $5 each. The 10th anniversary t-shirts are sold out.

EUUG Publications

The following EUUG publications may be ordered from the USENIX Association office.

The EUUG Newsletter, which is published four times a year, is available for $4 per copy or $16 for a full-year subscription. The earliest issue available is Volume 3, Number 4 (Winter 1983).

The July 1983 edition of the EUUG Micros Catalog is available for $8 per copy.

4.3BSD UNIX Manuals

The USENIX Association is sponsoring production of the 4.3BSD UNIX Manuals for its Institutional and Supporting members.† This article provides information on the contents, cost, and ordering of the manuals.

The 4.3BSD manual sets are significantly different from the 4.2BSD edition. Changes include many additional documents, better quality of reproductions, as well as a new and extensive index. All manuals are printed in a photo-reduced 6″×9″ format with individually colored and labeled plastic "GBC" bindings. All documents and manual pages have been freshly typeset and all manuals have "bleed tabs" and page headers and numbers to aid in the location of individual documents and manual sections.

A new Master Index has been created. It contains cross-references to all documents and manual pages contained within the other six volumes. The index was prepared with the aid of an "intelligent" automated indexing program from Thinking Machines Corp. along with considerable human intervention from Mark Seiden. Key words, phrases and concepts are referenced by abbreviated document name and page number.

While two of the manual sets contain three separate volumes, you may only order complete sets.

The costs shown below do not include applicable taxes or handling and shipping from the publisher in New Jersey, which will depend on the quantity ordered and the distance shipped. Those charges will be billed by the publisher (Howard Press).

Manuals are available now. To order, return a completed "4.3BSD Manual Reproduction Authorization and Order Form" to the USENIX office along with a check or purchase order for the cost of the manuals. You **must** be a USENIX Association Institutional or Supporting member. Checks and purchase orders should be made out to Howard Press. Orders will be forwarded to the publisher after license verification has been completed, and the manuals will be shipped to you directly from the publisher.

Manual	Cost*
User's Manual Set (3 volumes) User's Reference Manual User's Supplementary Documents Master Index	$25.00/set
Programmer's Manual Set (3 volumes) Programmer's Reference Manual Programmer's Supplementary Documents, Volume 1 Programmer's Supplementary Documents, Volume 2	$25.00/set
System Manager's Manual (1 volume)	$10.00

* Not including postage and handling or applicable taxes.

4.2BSD Manuals are No Longer Available

† Tom Ferrin of the University of California at San Francisco, a former member of the Board of Directors of the USENIX Association, has overseen the production of the 4.2 and 4.3BSD manuals.

4.3BSD Manual Reproduction Authorization and Order Form

This page may be duplicated for use as an order form

USENIX Member No.: _____

Purchase Order No.: _____

Date: _____

As the representative of a USENIX Association Institutional or Supporting Member in good standing, and as a *bona fide* license holder of both a 4.3BSD software license from the Regents of the University of California and a UNIX®/32V or System III or System V license or sublicense from AT&T and pursuant to the copyright notice as found on the rear of the cover page of the UNIX®/32V Programmer's Manual stating that

> "Holders of a UNIX®/32V software license are permitted to copy this document, or any portion of it, as necessary for licensed use of the software, provided this copyright notice and statement of permission are included,"

I hereby appoint the USENIX Association as my agent, to act on my behalf to duplicate and provide me with such copies of the Berkeley 4.3BSD Manuals as I may request.

Signed (authorized Institutional representative): _____

Institution: _____

Ship to:

Name: _____

_____

_____

_____

_____

Phone: _____

Billing address, if different:

Name: _____

_____

_____

_____

_____

Phone: _____

The prices below **do not** include shipping and handling charges or state or local taxes. All payments must be in US dollars drawn on a US bank.

4.3BSD User's Manual Set (3 vols.)	_____	at $25.00 each	= $_____
4.3BSD Programmer's Manual Set (3 vols.)	_____	at $25.00 each	= $_____
4.3BSD System Manager's Manual (1 vol.)	_____	at $10.00 each	= $_____
Total	_____		$_____

[ ] Purchase order enclosed; invoice required.
(Purchase orders **must** be enclosed with this order form.)

[ ] Check enclosed for the manuals: $_____
(Howard Press will send an invoice for the shipping and handling charges and applicable taxes.)

Make your check or purchase order out to Howard Press and mail it with this order form to:

Howard Press
c/o USENIX Association
P.O. Box 2299
Berkeley, CA 94710

for office use: l.v.: _____ check no.: _____ amt. rec'd: _____

USENIX Association
P.O. Box 2299
Berkeley, CA 94710

First Class Mail

FIRST CLASS MAIL
U.S. POSTAGE PAID
El Cerrito, CA 94530
Permit No. 87

RIACS Mail System

Go Tournament

System Administration Workshop

Call for Nominations

Change of Address Form

Please fill out and send the following form through the U.S. mail to the Association Office at the address above.

Name: _____ Member #: _____

OLD: _____

_____

_____

_____

NEW: _____

_____

_____

_____

Phone: _____

uucp: {decvax,ucbvax}! _____